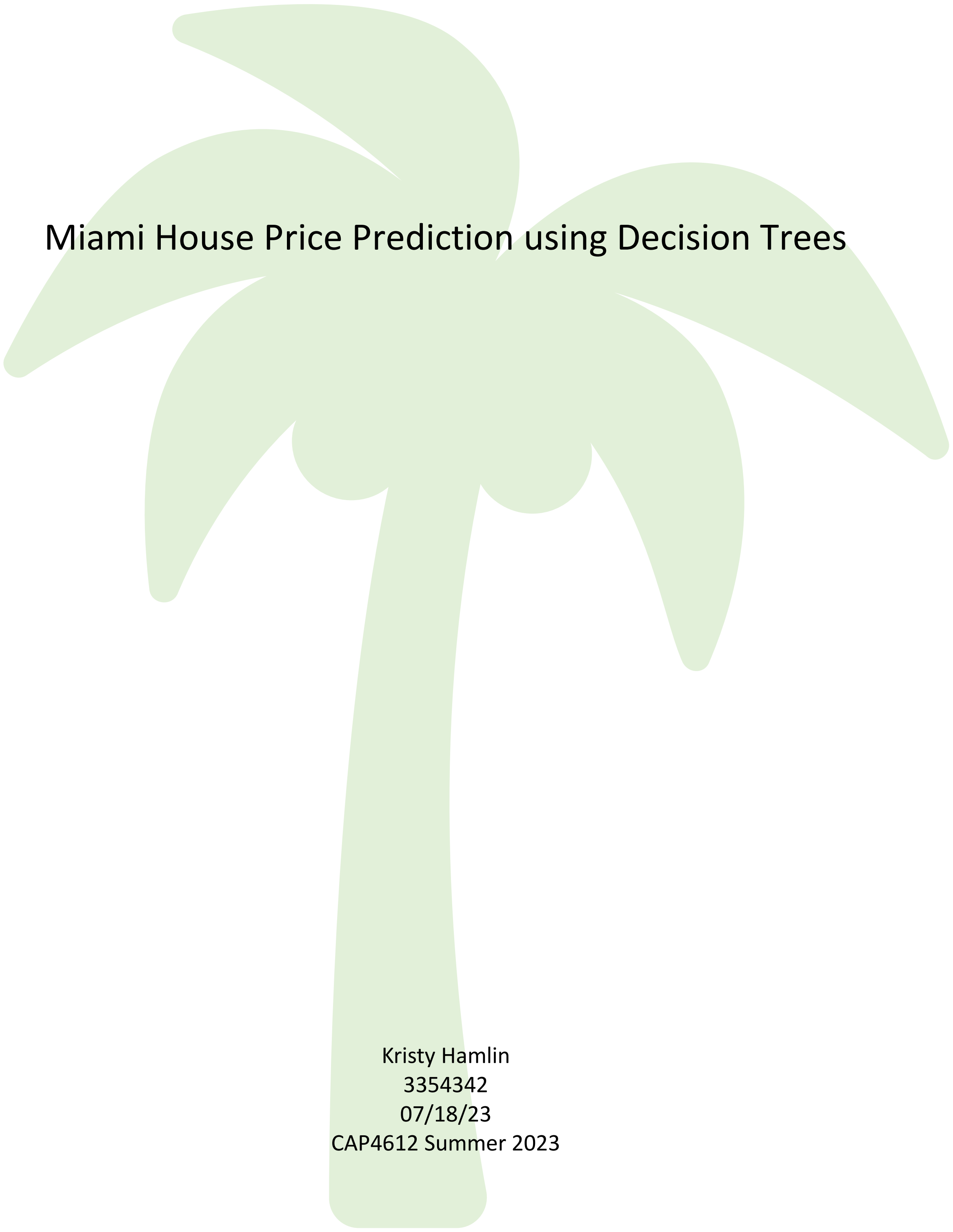


Miami House Price Prediction using Decision Trees

Kristy Hamlin
3354342
07/18/23
CAP4612 Summer 2023

Part I: Background

Introduction

Knowing how various factors impact house prices in our city is of clear value. When young professionals (like me?) fantasize about one day owning a home, the obvious constraint is money, and a plethora of variables come to mind that could affect house price: land area, floor area, distance to the ocean, distance to the city, the age of the home – to name a few. All these factors seem important, but which are most important, and what happens when multiple factors are at odds with one another? The purpose of my project is to utilize machine learning to be able to model the sale price of homes in Miami. To do this, I will use a dataset of almost 14,000 single-family homes sold in Miami in 2016. This dataset includes all the aforementioned fields and more.

This model could represent a powerful tool for home buyers. For example, if I am willing to compromise on the age of the home and distance to the city, would I be able to find a house of a reasonable price? Or, if I have evidence from the model that a home is overpriced, could this give me confidence in making a negotiation?

Related Work

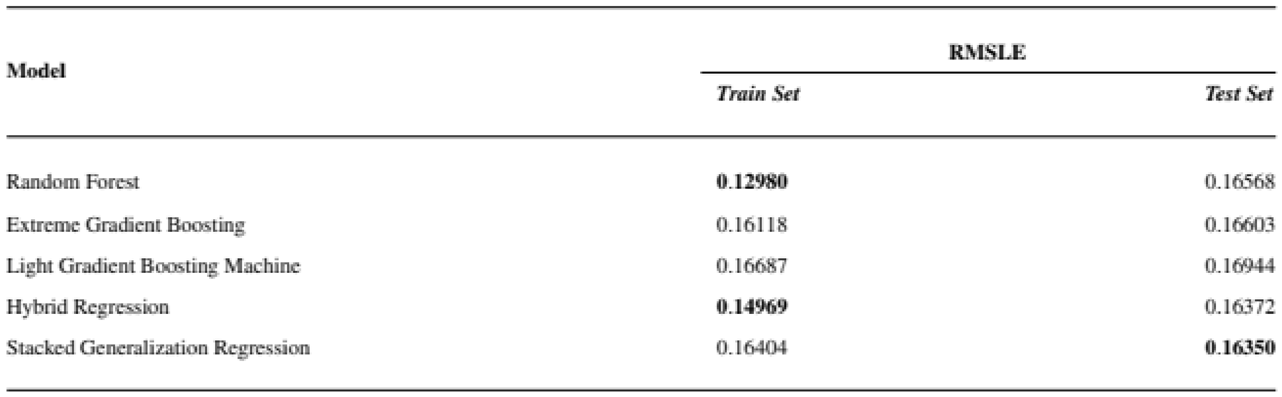
Although I genuinely created this semester project based on my own concerns, it is unsurprising that many others have already leveraged machine learning to predict house prices. In fact, in 2020 a team lead by Quang Truong compared alternate machine learning models in their ability to predict house prices, and therefore we will analyze their study to inform our research (Truong et al., 2020). The ”Housing Price in Beijing” dataset used by Truong includes over 300,000 objects with 26 variables, but one noticeable distinction from the Miami dataset is that the Beijing dataset spans 10 years, from 2009 to 2018.

Precisely as we learned in class, the Beijing data was preprocessed in a number of interesting ways, representing the first challenge overcome by the researchers. For example, variables that were missing from over 50% of the dataset were removed altogether. Ambiguous and mistranslated variables were cleaned; for example, the team writes that “bedrooms” were mistranslated to “living rooms”, and therefore the number of living rooms was set in a range from 1 to 4 (Truong et al., 2020, p. 434), although how the number between 1 and 4 was decided is not stated. Additionally, “contstructionTime” was replaced with “age” by using one to calculate the other. Moreover, outliers (defined using the inter-quartile range) were removed for all attributes (p. 434).

After preprocessing, Truong’s team was left with 19 attributes including longitude, latitude, district, distance from the center of Beijing, age of the house, floor area, and more (p 435). The team notes that the variables are not independent, by describing that they discovered a strong correlation between building age, building type, and distance from the center of Beijing (homes close to the center of Beijing are older but more expensive). Normalizing the data was a second challenge faced by the researchers: “numerical values were standardized, while categorical values were one-hot-encoded” (p 436). One-hot-encoding I had not heard of, but Lekhana Ganji writes that one-hot-encoding “is a technique that we use to represent categorical variables as numerical values….” (Ganji, para. 3) and additionally explains that one-hot-encoding takes every category and turns it into its own Boolean column, thereby increasing dimensionality. In this case, the use of one-hot-encoding increased the number of attributes from 19 to 58 (Truong et al., 2020, p 436).

Truong's team compared regression models and therefore used the Root Mean Squared Logarithmic Error to evaluate each model. They first used a Random Forest model, building k=900 alternate decision trees using bootstrap samples from the data, although they did not specify the number of values used for each bootstrap sample. Truong notes with marked enthusiasm that this model yielded "exceptional" (Truong et al., 2020, p. 437) results, with the lowest RMSLE on the training dataset among all methods. Extreme Gradient Boosting and Light Gradient Boosting, two more tree-based regression models, also performed well. Hybrid Regression, a model which combines different regression models, was used with 1/3 XGBoost, Random Forest, and LightGBM from the previous models, and interestingly performed somewhere in the middle of all 3 (p. 439). The final approach tested was Stacked Generalization, which performed similarly to the EGBoost and LightGBM methods alone (p. 439).

Below is an excerpt of Table 2 from the paper, showing the RMSLE values for each model (Truong et al, 2020, p. 440):



Model	RMSLE	
	Train Set	*Test Set*
Random Forest	**0.12980**	0.16568
Extreme Gradient Boosting	0.16118	0.16603
Light Gradient Boosting Machine	0.16687	0.16944
Hybrid Regression	**0.14969**	0.16372
Stacked Generalization Regression	0.16404	**0.16350**

Figure 1: Excerpt from Truong et al. 2020, showing the RMSLE values for each Machine Learning model used in their study. Lower RMSLE values indicate a more accurate model.

Data

The Miami Housing Dataset (Miami Housing Dataset | Kaggle) includes the sale prices of almost 14,000 homes in Miami in 2016. The dataset includes 17 attributes with the following types:

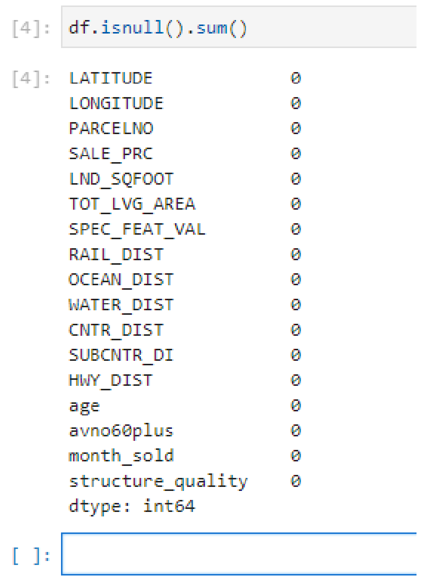
Table 1. Miami Housing Dataset Attributes

Name	Type	Unit	Description
PARCELNO	Nominal	N/A	Unique identifier for each property. About 1% appear multiple times.
SALE_PRC	Continuous	dollars	sale price of home
LND_SQFOOT	Discrete	square feet	Area of land, rounded to int
TOT_LVG_AREA	Discrete	square feet	Floor area of home, rounded to int
SPEC_FEAT_VAL	Continuous	dollar	Dollar value of "special" features such as pool
RAIL_DIST	Continuous	feet	Distance to nearest railway; used to estimate noise level
OCEAN_DIST	Continuous	feet	Distance to the ocean

WATER_DIST	Continuous	feet	Distance to nearest body of water
CNTR_DIST	Continuous	feet	Distance to Downtown Miami
SUBCNTR_DI	Continuous	feet	Distance to nearest subcenter of Miami
HWY_DIST	Continuous	feet	Distance to nearest highway; used to estimate noise level
age	Discrete	years	Age of home in years
avno60plus	Boolean	N/A	T if airplane noise exceeds acceptable level
structure_quality	categorical	N/A	Number from 1 to 5 indicating structural quality of the home
month_sold	categorical	N/A	The month in which the property was sold in 2016.
LATITUDE	continuous	degrees	Latitude of the property
LONGITUDE	continuous	degrees	Longitude of the property

The target value is, of course, sale price. A majority of the attributes in the dataset are continuous in nature, which should be useful for analysis. Additionally, the variables that are discrete (land area, living area, and age) may be able to effectively be treated as continuous due to their large range. Some of the other variables such as structure quality, month sold, and avno60plus may require additional experimentation or consideration for me to decide how best to handle them.

An initial exploration of the data using Jupyter (following instructions from the class lecture) reveals that none of the fields contain null values, impressively:



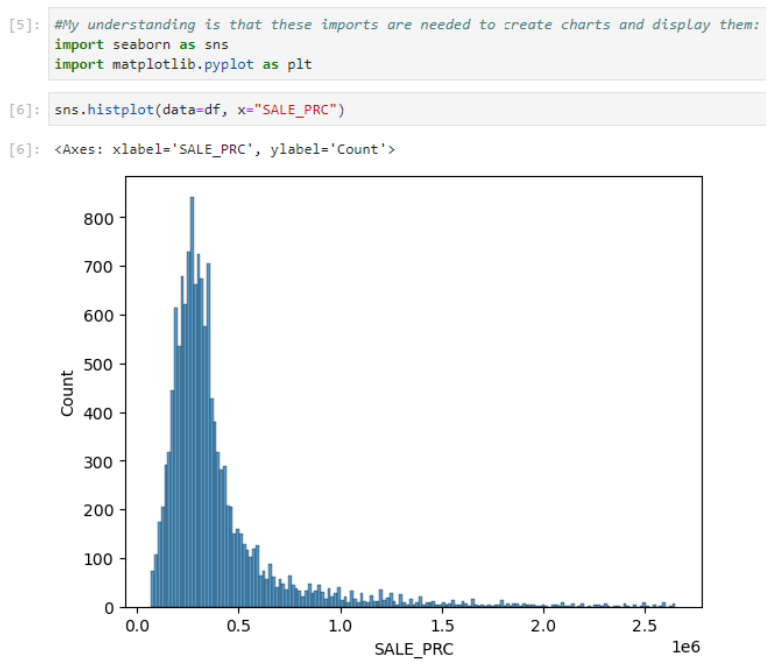
```
[4]: df.isnull().sum()

[4]: LATITUDE             0
     LONGITUDE            0
     PARCELNO             0
     SALE_PRC             0
     LND_SQFOOT           0
     TOT_LVG_AREA         0
     SPEC_FEAT_VAL        0
     RAIL_DIST            0
     OCEAN_DIST           0
     WATER_DIST           0
     CNTR_DIST            0
     SUBCNTR_DI           0
     HWY_DIST             0
     age                  0
     avno60plus           0
     month_sold           0
     structure_quality    0
     dtype: int64

[ ]:
```

Figure 2: Screenshot showing that no null values are detected in the Miami Housing Prices dataset.

Finally (to avoid this report becoming too long for the graders), we will take a look at the distribution of our target value, the home price, and note that it is (predictably), positively skewed:



```
[5]: #My understanding is that these imports are needed to create charts and display them:
     import seaborn as sns
     import matplotlib.pyplot as plt

[6]: sns.histplot(data=df, x="SALE_PRC")

[6]: <Axes: xlabel='SALE_PRC', ylabel='Count'>
```

Figure 2: Screenshot showing the histogram of house sale prices in the Miami Housing Prices dataset produced by Seaborn in Jupyter Notebook.

Part II: Implementation

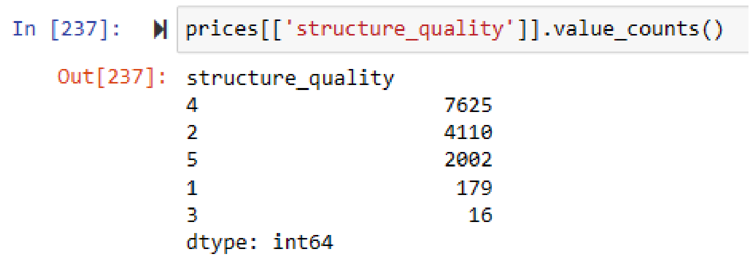
Libraries Used

In this project I relied on Pandas, matplotlib, Numpy, StandardScaler, train_test_split, and DecisionTreeRegressor (the last 3 are from sklearn). Pandas is used to import the datasets as dataframes and work with them. Matplotlib is used to make high-quality graphs from large amounts of data. NumPy is used for performing math with various array objects. Finally, StandardScaler, train_test_split, and the DecisionTreeRegressor from sklearn are used to perform data normalization, split the data into training and test data, and create the model respectively.

Data Exploration

Before manipulating the data, I used matplotlib to explore the data and get an idea of the best way to approach data preprocessing. The original dataset has 17 dimensions, and I was suspicious that some of them would be superfluous. No fields had null values (as mentioned above). In the following discussion, I will use the field name as it appears in the Kaggle dataset interchangeably with plain English for what the field represents. For example, I may refer to HWY_DIST as such, or as “distance to the highway”. If a field name is confusing and not deducible clearly from plain English as to which column it indicates, I will state a brief explanation indicating which column I am discussing.

Curiosity fortunately lead me to investigate the field structure_quality. Structure_quality was a categorical value from 1 to 5 indicating the “quality” of the home. I was interested in the distribution of these values, and calling .value_counts() quickly revealed some problems with this field:



```
In [237]:  prices[['structure_quality']].value_counts()

Out[237]:  structure_quality
           4                    7625
           2                    4110
           5                    2002
           1                     179
           3                      16
           dtype: int64
```

Figure 3: Distribution of Values in the structure_quality Column

Out of 14,000 data points, it seems absurd that only 16 houses would belong to the middle category of 3. (Perhaps if there were 16 houses in the top category I would assume these were the front row mansions on Tahiti Beach! But for the 16 houses to be in category 3 makes little sense.). Because I could not find documentation confirming whether a quality of 1 or 5 indicated good quality, I found the mean price of each group. This revealed something else strange: the smallest group with a quality of 3 also had a mean price much higher than the others. The values 1, 2, 4, and 4 show a steady increase in mean price from 162,000 to 743,000, but the 16 houses with a quality of 3 had a mean price of 1,800,000, over a million dollars more than the mean of the houses with a quality of 5.

Next, I proceeded to use matplotlib to produce scatter plots so I could visually determine whether the different columns correlated well with price.

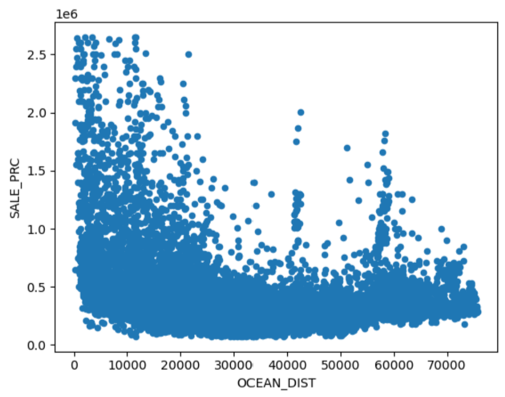


Figure 4: Price vs. OCEAN_DIST

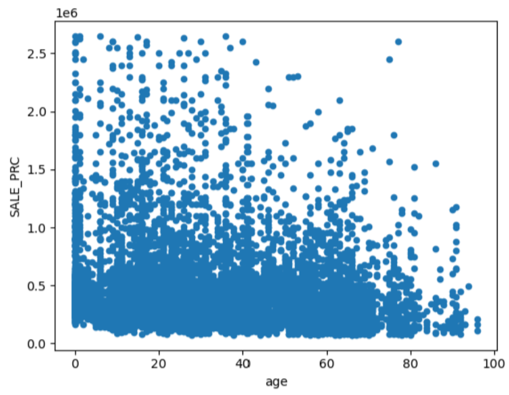


Figure 5: Price vs age

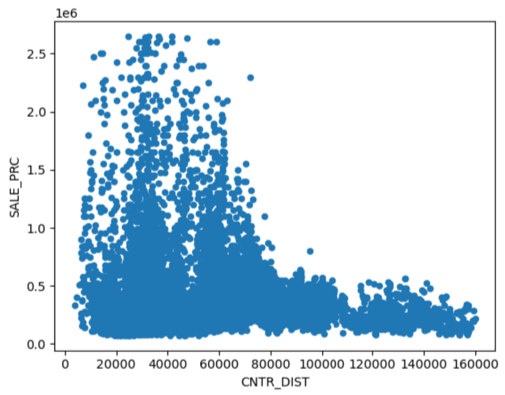


Figure 6: Price vs CNTR_DIST

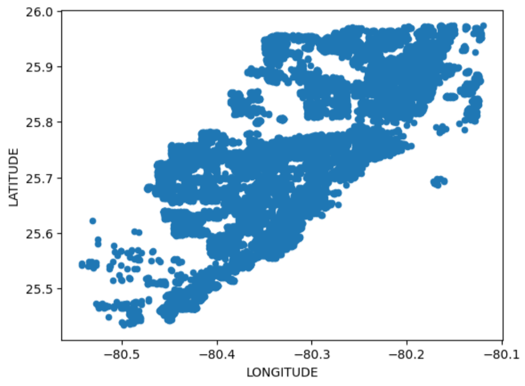


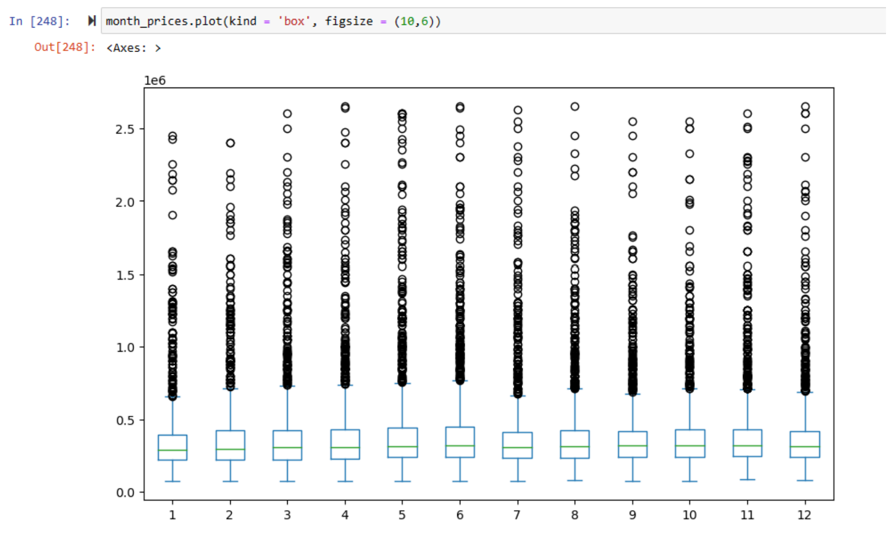
Figure 7: LATITUDE vs LONGITUDE

From Figures 4 – 7 we can see that, predictably, distance to the ocean and distance to the center of the city correlate clearly with price. Homes that are farther from the ocean and city cost less money (Again, this is why everyone hates West Kendall). Age also correlates with price in that there are very few houses that are very old and very expensive. I also plotted latitude and longitude to see visually where the houses were located in the city, and we can see that this plot clearly reveals the map of Miami, complete with some homes on Key Biscayne and South Beach.

Data Preprocessing

First, I intended to reduce the overall number of dimensions because I suspected that some of them were redundant or not informative. For example, Latitude and Longitude are likely redundant with distance to the city center (south is farther from downtown) and ocean distance (west is farther from the ocean) respectively. I therefore dropped LONGITUDE and LATITUDE.

I also suspected that the month_sold field was not informative, but I confirmed this before dropping the column by plotting the distribution of house prices for each month in 2016:



```
In [248]:  month_prices.plot(kind = 'box', figsize = (10,6))
Out[248]:  <Axes: >
```

Figure 8: Distribution of Sale Prices in Different Months

As we can see in Figure 8, the distribution of sale price is nearly identical in all 12 months (except perhaps that January and February seem to have fewer expensive outliers).

I suspected that avno60plus – a variable indicating that a home is very close to an airport and that there is noise – would not be essential. I likewise plotted the distribution in price: (0 indicates the home is not close to airport noise)

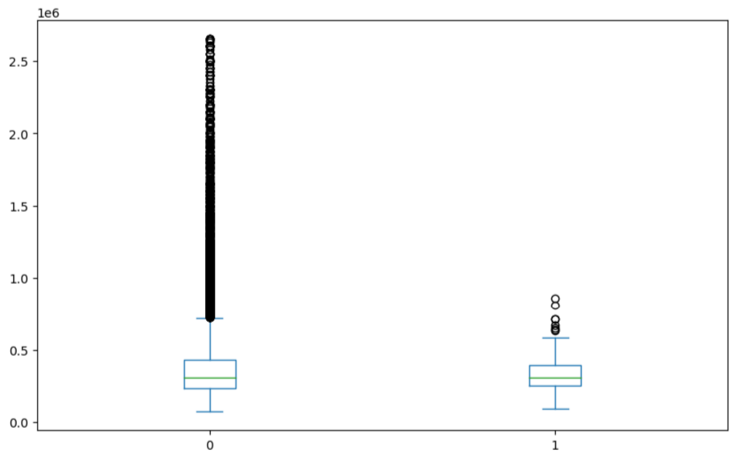


Figure 9: Distribution of Sale Prices in Homes close to Airports

Interestingly, the spread (not counting outliers) is extremely similar. The fact that the means, upper and lower quartiles are so similar was surprising. Therefore I proceeded to drop this column.

I plotted additional scatterplots to decide which fields to keep:

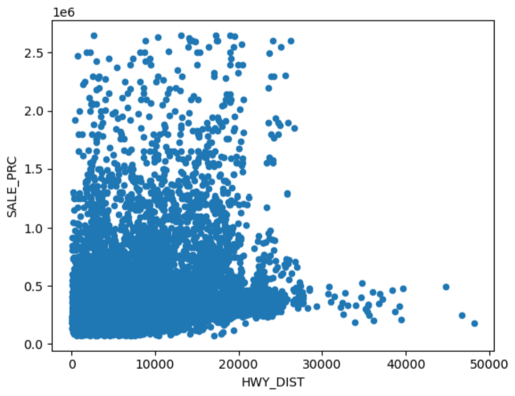


Figure 10: Sale Price vs HWY_DIST

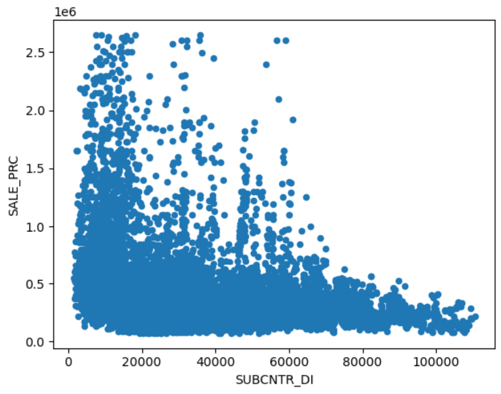


Figure 11: Sale Price vs SUBCNTR_DI

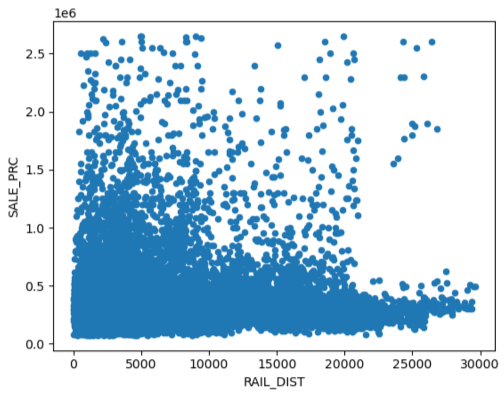


Figure 12: Sale Price vs RAIL_DIST

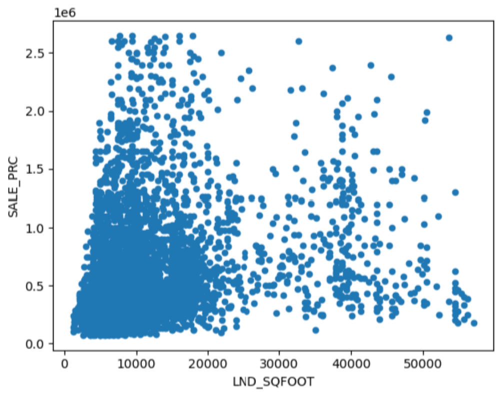


Figure 13: Sale Price vs LND_SQFOOT

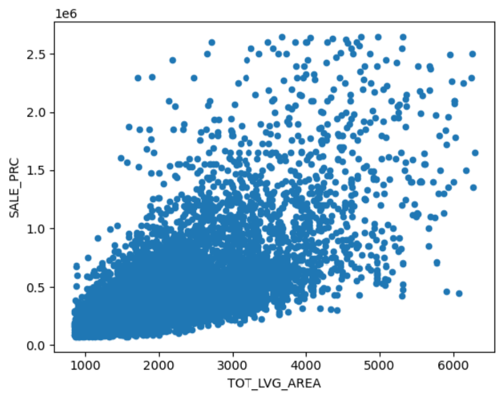


Figure 14: Sale Price vs TOT_LVG_AREA

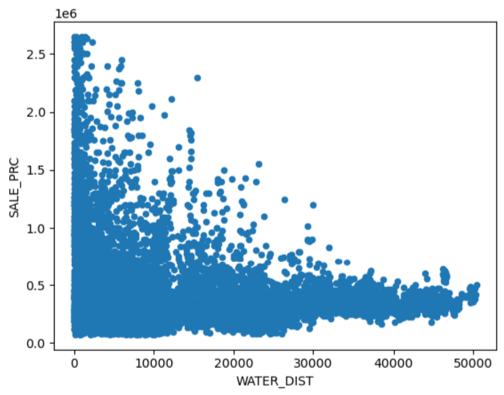


Figure 15: Sale Price vs WATER_DIST

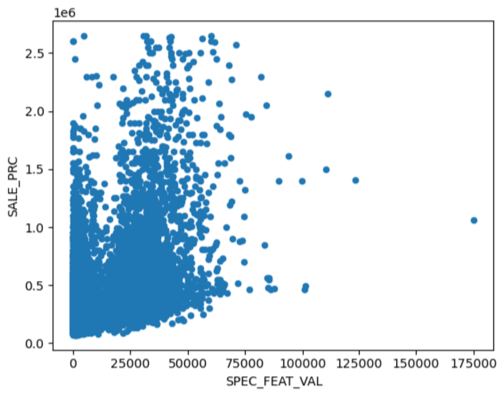


Figure 16: Sale Price vs SPEC_FEAT_VAL

After examining the scatter plots, I dropped HWY_DIST and kept the other fields. HWY_DIST seemed to be the least correlated to price on a visual basis.

As the second step in my preprocessing pipeline, I removed all outliers in all remaining fields. The first group of "outliers" I removed was the group of 16 rows with a structure quality of 3, since these homes were highly unusual in that their mean price was over a million dollars more expensive than the houses with the highest structure quality of 5, and this group also broke the pattern of a steady increase in price with structure quality.

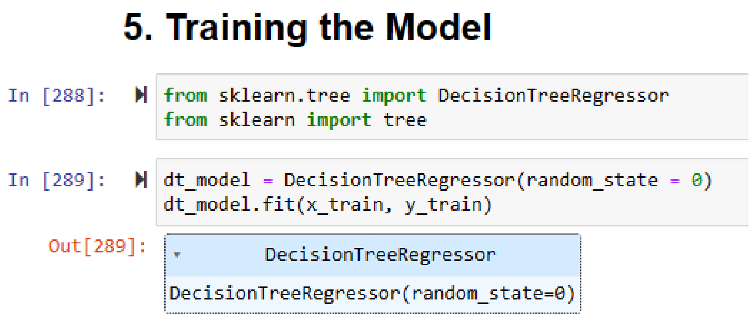
Next, I proceeded to remove all rows that had an outlier in any field as defined by the standard deviation of the field. For each field, I calculated the mean, and upper and lower limits defined by three standard deviations on either side. Then, I removed all rows that had values beyond those limits. I chose to do this because Truong's (2020) team removed outliers in all fields and experienced great success predicting housing prices in Beijing using decision tree models, as discussed previously.

After closing and opening my Jupyter Notebook, I started receiving warnings when re-running the area that drops outliers. However, I used printout statements to confirm the presence of outliers before their removal and their absence afterward for each field, so I believe I can safely ignore these warnings. There were no lower outliers detected in any fields, which is expected because the lower limits calculated were always negative.

Finally, I normalized every field except price and structure quality. I wanted to keep price in dollars for interpretability. I used the StandardScaler from sklearn.preprocessing. In brief, I created a copy of the dataframe and then systematically calculated the standardized values for each column and replaced the original values in the copy of the dataframe. I also called the .head() method to look at the values and make sure that they looked as expected after being standardized.

Training the Model

After preprocessing the data, it was almost time to create the model. Before doing so, I separated the dependent and independent variables into separate dataframes, and then used train_test_split from sklearn.model_selection to separate the data into 20% test data and 80% training data. Finally, I imported the DecisionTreeRegressor from sklearn.tree and used .fit() and the training dataframes to train the model:



5. Training the Model

```
In [288]:  from sklearn.tree import DecisionTreeRegressor
           from sklearn import tree

In [289]:  dt_model = DecisionTreeRegressor(random_state = 0)
           dt_model.fit(x_train, y_train)

Out[289]:          DecisionTreeRegressor
           DecisionTreeRegressor(random_state=0)
```

Figure 17: A Screenshot from Jupyter Notebooks showing the creation of the Decision Tree Model

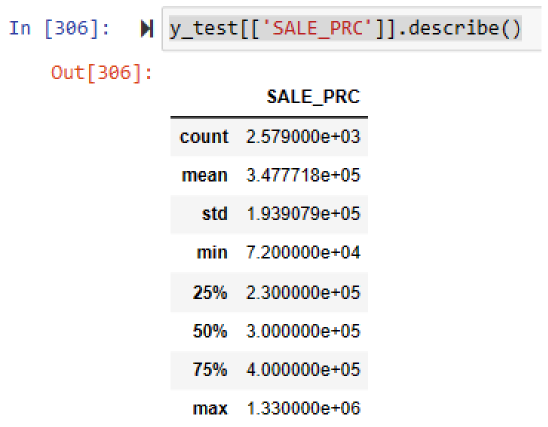
Part III: Analysis and Conclusion

Results

Because I used a regression model, accuracy, precision, and F1 scores are not applicable. Instead, I inspected the R-squared value of the regression model and the mean absolute error. I prefer the mean absolute error because it is easily interpretable; it maintains the same units as the dependent variable, and we intuitively understand that it is the average deviation between all the predictions and all the actual values.

The mean absolute error for my model on the testing data was 52,870. Initially, I was disappointed by how far off this seemed to be. However, I decided to take a closer look at the mean and standard deviation in price of the testing dataset (Figure 18). The mean house price was 347,700, and the standard deviation was 193,900. With these values in mind, the mean absolute error of 52,870 is only a fraction of 0.273 of the standard deviation. This is not extremely accurate, but maybe for a one-semester undergraduate project it is acceptable. It seems to indicate that the model at least is in the right ballpark in its predictions.

Moreover, the R-squared value was 0.78. Since the maximal value possible is 1, this is a pretty good score.



```
In [306]:  y_test[['SALE_PRC']].describe()

Out[306]:
```

	SALE_PRC
count	2.579000e+03
mean	3.477718e+05
std	1.939079e+05
min	7.200000e+04
25%	2.300000e+05
50%	3.000000e+05
75%	4.000000e+05
max	1.330000e+06

Figure 18: Summary Statistics for the Testing Dataset Price Values

Discussion

What did we learn from this exercise? Will I ever own a house? What are the most important areas to be willing to compromise in for a broke Miami native? Fortunately, the DecisionTreeRegressor from sklearn.tree allows us to take a look at the relative importance of each feature with the feature_importances_ attribute (discussed in EvidenceN, 2021). I sorted and plotted the relative importance of the features to take a closer look:

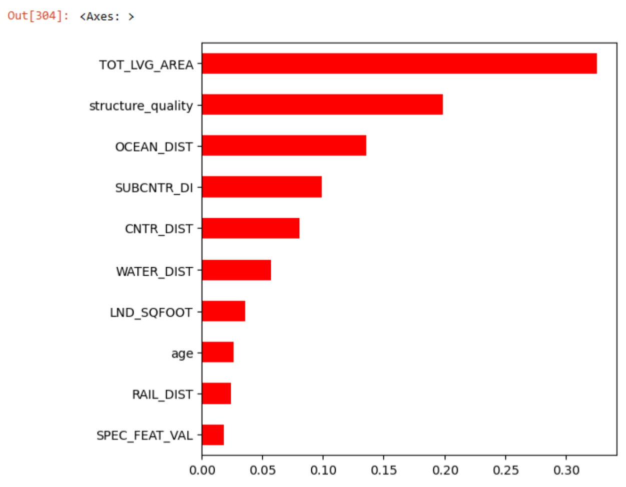


Figure 19: Relative Importance of Independent Variables

Fascinatingly, the highest predictor of price was total living area and structure quality. These factors were more important than the distance to the ocean, which I found astounding. This dataset, to my understanding, did not include apartment buildings, so the presence of non-millionaire apartments on South Beach should not be dragging down this feature in importance. Interestingly, distance to the city is only the 5th most influential factor, meaning I may not need to move to West Kendall (thank god). The difference in importance between land square footage and living area is very surprising, and suggests that with funds, one could buy a large property with a small house relatively cheaply and then build a bigger house on it. Alternatively, one could buy a house far from the city center or nearest sub-center, then move it closer. (That one was a joke.)

Future Work

It is not possible to compare my model to Truong's model for accuracy, since he used a the RMSLE to judge the models. Moreover, it does not say in his paper what unit his results are in, and I am not sure if price was also z-standardized along with his input.

Given more time, it would be fascinating to compare different techniques and models to see if I could improve the MAE I achieved here. Specifically, I would first try including more of the original fields, or different approaches to dealing with the one categorical variable, structure quality. Truong's team (2020) handled categorical variables using one-hot-encoding, but because my research of decision trees did not indicate this was strictly necessary and I was already spending a lot of time on preprocessing, I decided not to pursue this route. Moreover, given that Z-score standardization may not be needed for decision tree models, I would be interested to see how different the MAE would be by simply not performing standardization.

Finally, it would be interesting to compare Decision Trees to other predictive methods. Particularly, the part of my brain that is highly amused by word-humor wants to employ the KNN regressor to see if the literal nearest neighbors to any given home (by using only the latitude and longitude fields) provide a good estimate of price. Given that houses near each other tend to have similar age, quality, and many other factors, and are by definition the same distance to the city, water, etc, this might perform well! However, drawing helpful conclusions from such a model would be more difficult – "To find an affordable house, look for a house near other affordable houses!" But it would still be entertaining.

In conclusion, I have drawn some very interesting and helpful insights from this model. I may actually keep in mind what I have learned about the relative influence of the different fields when moving out!

REFERENCES

EvidenceN. (2021) How to Interpret Decision Tree Regressor Model Resutls in Python, Scikit-Learn, Matplotlib. Retrieved July 17, 2023 from https://youtu.be/z9GDlJs1qDA

Ganji, L. (2023, April 18). *One hot encoding in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/

Krishnan, Sandhya. (2021). Decision Tree for Classification, Entropy, and Information Gain. Retrieved July 16, 2023. from https://medium.com/codex/decision-tree-for-classification-entropy-and-information-gain-cd9f99a26e0d

Nwanganga, F. (2021). Machine Learning with Python: Foundations [Course]. *LinkedIn Learning*. https://www.linkedin.com/learning/machine-learning-with-python-foundations

Truong, Q., Nguyen, M., Dang, H., & Mei, B. (2020). Housing price prediction via Improved Machine Learning Techniques. *Procedia Computer Science*, *174*, 433–442. https://doi.org/10.1016/j.procs.2020.06.111

Unknown. (No Date) Miami Housing Dataset. Retrieved June 29, 2023. from https://www.kaggle.com/datasets/deepcontractor/miami-housing-dataset.